

Modern Dağıtık Dosya Sistemlerinin Yapısal Karşılaştırılması

Bahadır KARASULU, Serdar KORUKOĞLU

Ege Üniversitesi, Bilgisayar Mühendisliği Bölümü, 35100, Bornova, İzmir
{bahadir.karasulu, serdar.korukoglu}@ege.edu.tr

Özet: Günümüz dağıtık dosya sistemleri, dosya sistemi kaynaklarının hem yerel hemde geniş alan ağlarında etkin olarak paylaşılmasını sağlayarak dağıtık uygulamaların bu sistem üzerinden çalıştırılmasına olanak sağlar. Bu sistemler arasında AFS (Andrew Dosya Sistemi), NFS (Ağ Dosya Sistemi), Plan 9, CODA gibi dağıtık dosya sistemleri bulunmaktadır. Dağıtık dosya sistemlerine ait bazı sınıflandırmalar için farklı altyapılar mevcuttur. Bu sınıflandırmalar çoğu dosya sisteminin hataya dayanıklı çalışmaya uygunluğuna göre oluşturulmuştur. Çalışmamızda bu tarz günümüzde oldukça fazla kullanımda olan bazı dağıtık dosya sistemlerinin yapısal olarak birbirlerine göre üstün veya eksik yönlerinin detayları incelenmiş, sonuçlar tartışılmıştır.

Anahtar Kelimeler: Dağıtık dosya sistemi, Ağ dosya sistemi, Dağıtık sistem, Hataya dayanıklılık, İşletim sistemi.

Structural Comparison of Modern Distributed File Systems

Abstract: Distributed file systems provide effective sharing of file system resources by local area as well as wide area networks and distributed applications can run on these file systems. AFS (Andrew File system), NFS (Network File System), Plan 9, CODA and etc are some examples of these distributed file systems. Different infrastructures are available for some classifications which are owned by distributed file systems. Many of these classifications are built up, based on the capability of the file system to work in a fault-tolerance manner. In this work, we analyze superior and inferior properties of some distributed file systems by comparison and discussed the results in details.

Keywords: Distributed file system, Network file system, Distributed system, Fault tolerance, and Operating system.

1. Giriş

Bir dağıtık dosya sistemi, dosyaları sunucu denilen bir veya daha fazla bilgisayar üzerinde depolayarak, istemci denilen diğer bilgisayarların bu dosyalara kendi yerel dosyaları gibi erişilebilmelerini sağlayan ağ tabanlı dosya sistemidir. Dağıtık dosya sisteminde sunucu, istemciler tarafından çok maliyetli ve pratikte mümkün olmayan büyüklükte depolama alanları sunabilmektedir. Bir ağ üzerinden çalışma-

bilecek bir dosya sistemi dosyaların, yazıcı gibi kalıcı kaynakların ve depolama kaynaklarının bilgisayar ağları üzerinden paylaşılmasını sağlamalıdır. Bu amaçla 1970'li yılların ortalarında dağıtık dosya sistemleri geliştirilmeye başlanmıştır. İlk olarak, Svobodova [1]'nin çalışmasında belirttiği gibi istemciler için FTP-benzeri bir yaklaşım sunan Datacomputer kullanılmış fakat yetersiz depolama alanı nedeniyle yerini geçici dosya sistemi'ne bırakmıştır. İlerleyen yıllarda çeşitli dağıtık dosya sistemleri kulla-

nıma girmiştir. Bunlar arasında Svobodova [1] ve Swinehart vd. [2] çalışmalarında bahsedilen WoodstockFS, XDfs, SWALLOW, LOCUS, ACORN ve CMU (Carnegie Mellon Üniversitesi) 'ya ait VICE sayılabilir. Günümüze yaklaştıkça birçok başlıca dağıtık dosya sistemleri kullanıma girmiştir (Sun Microsystems'e ait Ağ Dosya Sistemi (NFS) ve CMU'ya ait Andrew Dosya Sistemi (AFS) gibi). Sınıflandırma yapılmak istenilirse bunların hataya dayanıklılıkları göz önüne alınarak ayırım yapmak mümkündür. Bu çalışmada belirli başlıca yaygın dağıtık dosya sistemleri ele alınarak güncel bazı dağıtık dosya sistemleri yapısal ve güvenlik açısından incelenme ve karşılaştırmaları yapılmıştır.

2. Dağıtık Dosya Sistemlerinin Genel Yapısı

Dağıtık dosya sistemleri, birçok yönetsel avantajı da beraberinde getirir. Erişim konusundaki bazı olumsuzlukların aşılması, kullanılacak dağıtık dosya sisteminin saydamlığının seviyesine bağlıdır [3].

Dağıtık dosya sistemlerine hataya dayanıklı olmaları açısından bakılırsa, bu tarz sistemler düğüm makineler arasında veri kopyalaması sırasında hataya karşı bulunan zafiyeti en aza indirerek, yüksek elde edilebilirlik ve çevrimdışı (bağlantı kopartılmışken) durumda bile işlemlerin devam edilebilirliğini sağlar. Bu tarz sistemlere en uygun örnekler CMU'ya ait CODA ve Microsoft Corporation'a ait MS-Dfs (Dağıtık Dosya Sistemi)'dir [4]. Bunların haricinde paralel çalışmaya uygun dağıtık dosya sistemleri, veriyi birden çok sunucuya yüksek başarımla elde edilebilmesi için dağıtabilmektedirler. Bunlar Yüksek Başarımlı Hesaplama (HPC) alanında kullanılırlar [4]. Ayrıca bazı dağıtık dosya sistemleri hem paralel hem de hataya dayanıklı yapıya uygundur. Bunlar yüksek başarımlı hesaplama ve yüksek elde edilebilirliğe sahip kümelerde kullanılmaktadırlar, en uygun örnekleri arasında

Google firmasının geliştirdiği Google Dosya Sistemi (GoogleFS) bulunmaktadır.

3. Yaygın Dağıtık Dosya Sistemleri

Çeşitli dağıtık sistemler arasında özellikle ön plana çıkanlar arasında, Bell laboratuvarları'na ait (ACL bulundurmayan) Plan 9 (9P) ve Dağıtık Hesaplama Ortamı/Dağıtık Dosya Sistemi (DCE/DFS), şu an IBM 'in elindeki Andrew Dosya Sistemi (AFS) ve Sun Microsystems'e ait Ağ Dosya Sistemi (NFS) ve CMU'nun geliştirdiği mobil kullanıcıların zaman zaman ağ'dan çıkıp tekrar girebilmelerine olanak tanıyan CODA dağıtık dosya sistemi bulunmaktadır [5]. Hataya dayanıklılığın sağlanmasında, dağıtık dosya sistemi gereksiz bazı bileşenleri ve durumları da içerebilmektedir. Bunların giderilmesine yönelik çalışmalar sonucu dağıtık dosya sistemleri çeşitlenmektedir. Diğer sistemlere göre daha yaygın olarak kullanılan dağıtık dosya sistemleri aşağıda incelenmektedir.

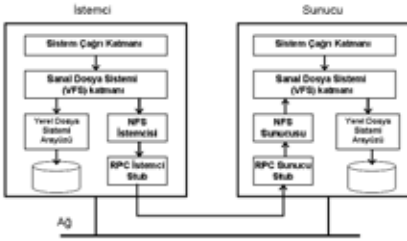
3.1. NFS Çalışma Mekanizması

UNIX sistemlerde geniş çapta kullanılan NFS, Sun Microsystem tarafından 1985 yılında ilk kez NFS istemci ve sunucuları arasında harici veri sunumu için bir RPC (Uzak Yordam Çağrısı) arayüz tanımlama protokolü olarak tasarlanmıştır. İlk üç versiyon'un detayları Callaghan'ın [6] çalışmasında mevcuttur. NFS sunucuları, durumsuz (stateless) olarak nitelendirilir. Bunun nedeni bu tarz sunucunun kendi dosyalarına erişen istemcilerin durumları hakkında bilgiyi depolamamasıdır. Bu arada istemciler kendi yerel önbelleklerindeki dosyaları veya dizinler için farklı ve çatışan kopyalara sahip olabilmektedir [7]. Her bir istemcinin isim uzayı farklıdır.

3.1.1 Mimari ve İletişim

NFS'nin altında yatan model, uzak dosya servisi'dir. Bu model ile istemcilere uzaktaki bir sunucu tarafından yönetilen dosya sistemi-ne saydam bir erişim önerilmektedir. İstem-

çiler aslında dosyanın gerçek yerleşiminden habersizlerdir. Dosya sistemine erişmek isteyen bir istemcinin ilk önce kendi yerel işletim sistemine sistem çağrılarını ile erişmesi gerekir, bunu yaparken dağıtık dosya sistemleri için *de facto* standart olan Sanal Dosya Sistemi (VFS) arayüzünü kullanması gerekir. Bu sayede NFS istemcisi bileşenine işlemler devredilir. Hem istemcide hemde sunucuda *stub* adı verilen arayüzler bulunmaktadır. İstemci mesajını RPC-istemci *stub*'a verir. İstemci-sunucu iletişimi RPC'ler ile yapılır. NFS sunucusu ise gelen istemci isteklerine cevap verir. RPC-sunucu *stub* gelen mesajları açar, böylece NFS sunucusu bunları düzgün biçimde VFS dosya işlemleri şekline dönüştürür. Daha sonra VFS katmanını sayesinde ilgili yerel dosya sistem arayüzü üzerinden bu işlemler gerçekleştirilir. Şekil 1 'de bu işleme ait adımlar gösterilmektedir.

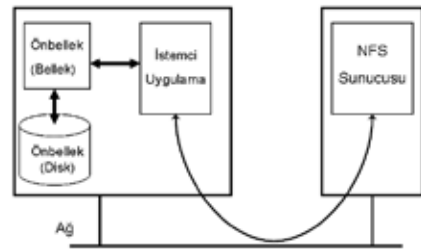


Şekil 1. UNIX sistemler için temel NFS mimarisi.

NFS 'de bir sunucu ile bir istemci arasındaki tüm iletişim Açık Ağ Hesaplama RPC (ONC RPC) protokolü ile yapılmaktadır [8, 9]. NFS v4, birkaç RPC'nin tek bir isteğe gruplanabildiği bileşen prosedürlerini desteklemektedir [5]. NFS için protokol ile istemci-sunucu etkileşiminde v2 ve v3'e kadar durumsuz (stateless) sunucu yaklaşımını önerilirken, NFS v4'te bundan vazgeçilmiştir. Bu durumun tek avantajı basit oluşudur. NFS v4'te durumlu (stateful) sunucu tipine geçilmiştir.

3.1.2 İsimlendirme ve Senkronizasyon

NFS için isimlendirme önemli bir konudur. Buradaki temel amaç istemcinin, uzak dosya sistemine tamamen saydam olarak erişebilmesidir. Bu saydamlık istemcinin yerel dosya sistemini uzak dosya sistemine bağlaması yoluyla gerçekleştirilir. Tüm uzak dosya sistemini bağlamak yerine, ilgili parçanın bağlanmasının daha etkin olacağı göz önüne alınmıştır [5]. NFS için, istemci makinesinde ayrı bir süreç ile çalıştırılan Otomatik Bağlayıcı (Automounter) sayesinde uzak bir dosya sistemini anında bağlamak mümkün olmaktadır. Dosyalar için belirli sayıda öznitelik NFS tarafından desteklenmektedir. En önemli öznitelik ACL (Erişim Kontrol Listesi) 'dir ve UNIX tarzı dosya erişim izinlerini destekler. Dağıtık dosya sistemlerinde tutarlılık senkronizasyonu önemli bir konudur. Dosyalar merkezi bir sunucuda tutulursa tutarlılığı sağlamak görece kolaydır, fakat bu performans problemlerine yol açar. Dosya kapatma sırasında güncel veri'nin sunucuya gönderilmesi temeline dayalı oturum semantik kuralı çoğu dağıtık dosya sisteminde olduğu gibi NFS'de de bulunur. Bu upload/download temelli bir modeldir. Dosya üzerinde en son değişiklik yapan en güncel veriyi sunucuya göndermiş olur. Şekil 2'de görüldüğü gibi istemci uygulama NFS sunucusuna ve kendi önbelleklerine (bellek ve disk) bağlıdır ve ağ üzerinden NFS sunucusuna erişerek dosya paylaşımı yapabilmektedir [10].



Şekil 2. NFS'de önbelleklemeli bir istemci ile NFS sunucu arası iletişim.

Bunların haricinde dosya erişimlerinde dosya kilitleme mekanizmalarında sorun yaşanmaması için NFS’de ayrı bir protokol ile idare edilen bir kilitleme mekanizması yöneticisi (stateful biçimde) bulunur. NFS’de bir istemci daha önce (kısmen) önbelleklemediği ve önceden kapalı olan bir dosyayı açmak isterse elindeki önbelleklenmiş veriyi tekrar geçeriiletmesi (revalidate) gerekir. NFS v4’te, dosya replikasyonuna en az destek verilirken, tüm sistemin replika edilmesine olanak sağlanmaktadır [10].

3.1.3 Hataya karşı dayanıklılık

NFS sistemi temelde durumsuz (stateless) sunucuyu desteklediği için, her RPC iletişim aşamasında sorunla karşılaşmak olasıdır. Herhangi bir sistem çökmesinde bir durum kaybı olmamaktadır, fakat (stateful) kilitleme yöneticileri istemci durumları hakkında bilgi sahibi olmak isterler. RPC isteklerinin kaybolması durumunda tekrar tekrar gönderilmesi bir problem oluşturduğu için, sunucu- tarafında çift kez gönderilmiş istek önbelleği kullanımı ve her bir RPC isteğinin başlık kısmına tekil bir işlem tanımlayıcı atanması ile bu sorun halledilmiştir [10].

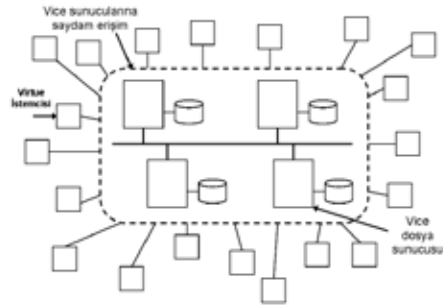
3.2 CODA Çalışma Mekanizması

CMU tarafından 1990’larda geliştirilen ve şu an birçok UNIX tabanlı sistemde kullanılan bir dağıtık dosya sistemi olan CODA, NFS’nin aksine yüksek dereceli elde edilebilirliği ana amaç olarak belirlemiştir. Bu yüzden önbellekleme yapmış bir istemci için uygun şema sayesinde sunucu ile bağlantı kopmuş olsa bile ilgili istemci dosya üstünde değişiklik yapmaya devam edebilmektedir. CODA yüksek dereceli ölçeklenebilirlik ve güvenlik sağlamaya çalışır. AFS versiyon 2’nin takipçisi olarak CODA geliştirilmiştir [11]. AFS tasarlanırken on bin’in üzerinde iş istasyonunun sisteme erişebileceği göz önüne alınarak oluşturulmuş ve iki gruba bölünmüş-

tür. İlk grup görece küçük sayıda ve merkezi olarak yönetilen adanmış Vice dosya sunucularını içerirken, ikinci grup ise çok sayıda Virtue iş istasyonlarını (bağlantı yapmak için gerekli) kullanıcıların ve süreçlerin dosya sistemine erişim yapabilmeleri için içermektedir.

3.2.1 Mimari ve İletişim

CODA, AFS ile aynı organizasyona sahiptir. Her bir Virtue iş istasyonu makinesi Venus isimli kullanıcı-seviyesi sürece sahiptir ve bunun görevi NFS’deki istemciye benzerdir. Venus süreci, Vice dosya sunucusu tarafından bulundurulmuş dosyalara erişimi sağlamaktan sorumludur. Venus, dosya sunucusuna erişimin mümkün olmadığı durumda da istemcinin işlemlerine devam edebilmesine olanak tanır [3]. VFS ile yapılan organizasyon NFS’ninkine benzemektedir. İstemcilerine CODA, geleneksel UNIX-tabanlı dosya sistemi gibi davranmaktadır. NFS’nin aksine CODA, Vice sunucuları tarafından kontrol edilen global paylaşımlı isim uzayını sağlamaktadır [11]. Şekil 3’te CODA’nın temel aldığı AFS sisteminin tüm organizasyonu (kaba bir görünüm ile) görülmektedir.



Şekil 3. AFS (CODA'ya temel teşkil eden) sisteminin tüm organizasyonu.

CODA’da RPC ile iletişim yerine RPC2 ile iletişim gerçekleştirilmektedir. RPC2’nin en önemli özelliği yan etkileri desteklemesidir [8]. Yan etki (side effect) bir istemci ile bir sunucunun, uygulamaya-özümlü bir protokol

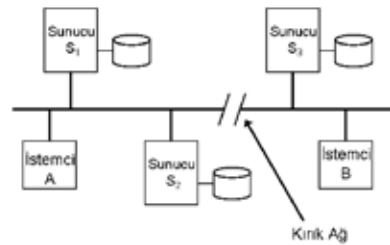
kullanarak iletişime geçebilecekleri bir mekanizmadır. Ayrıca RPC2 multicasting'i de desteklemektedir. Bu özellik onu diğer RPC sistemlerden ayırır. RPC2 paketinin bir parçası olarak paralel (eşzamanlı) RPC mesajlaşması gerçekleştirilebilmektedir, buna **MultiRPC** denilmektedir. Bunun anlamı, bir çağrıyı yapmanın, birden çok alıcıya (her bir alıcıya) istek gönderebilmesidir. CODA için istemci süreçler Venus süreçleri, sunucu süreçler ise Vice süreçleri olarak kesinlikle ayrılmışlardır [12].

3.2.2 İsimlendirme ve Senkronizasyon

CODA, UNIX benzeri bir isimlendirme sistemi bulundurur. Dosyalar gruplanarak hacimlere ayrılmıştır. Bu hacimler UNIX'teki disk bölümlerine benzemektedir ve onlar gibi mount (bağlama) edilebilirler. Genelde bir hacim bir kullanıcı ile ilgili dosya topluluklarını bulundurmaktadır. Hacimler CODA için önemlidir, çünkü tüm isim uzayının oluşturulduğu temel birimlerdir [3]. Bu mekanizma NFS v4'üne benzemektedir. Paylaşımlı dosyaların topluluğunun birden çok Vice dosya sunucusu arasında dağıtıldığı ve replika edildiğini göz önüne alırsak, her bir dosyanın tekil olarak tanımlanması önemlidir. Bu tanımlama, replikasyon ve yerleşim saydamlığı sağlanabilir halde iken dosyanın fiziksel konumuna kadar onun takip edilebilmesini içerir. CODA'daki her dosya tam olarak bir hacim tarafından içerilmektedir. Hacimler sunucular arasında replika edilebilirler. Bu yüzden CODA'da mantıksal ve fiziksel hacimler arasında bir ayrım yapılmıştır [12].

CODA, işlemsel semantik sağlamaya çalışır. CODA'nın asıl uğraştığı problem, oldukça büyük dağıtık dosya sisteminde, dosya sunucularının bazılarının veya tümünün geçici olarak ulaşılamaz olduğu durumdur. Bu durum, sunucu veya ağda karşılaşılan bir hata olabileceği gibi, ağ servisinden kasten çıkan (bağlantısını koparan) bir mobil istemciden de kaynaklanmış olabilmektedir. İstemci-tarafli önbellekleme CODA için iki sebepten dolayı

önemlidir. İlk sebep, AFS'de de izlenen yaklaşımda olduğu gibi ölçeklenebilirliğin başarılabilmesi için önbellekleme yapılmasıdır [13]. İkinci sebep, istemciyi sunucunun elde edilebilirliğine daha az bağımlı hale getiren hataya dayanıklılığın yüksek derecesinin başarılanmasını sağlayan konunun önbellekleme oluşudur [14]. Bu sebepten CODA'daki istemciler tüm dosyaları önbelleklerle. Bir dosya okunmak veya yazılmak üzere açıldığında, dosyanın bütün bir kopyası istemciye transfer edilerek orada depolanır. CODA, dosya sunucularının replika edilmesine izin verir. Bir hacmin bir kopyasına sahip sunuculardan oluşan topluluğa hacmin Hacim Depolama Grubu (VSG) denilir. Hata oluştuğunda, istemci hacmin VSG'indeki tüm sunuculara erişemeyebilir. Bir istemci'nin Erişilebilir Hacim Depolama Grubu (AVSG), istemcinin iletişimde olduğu sunuculardan (VSG'deki) oluşur. Eğer AVSG boş ise, istemcinin bağlantısını kopardığı söylenilebilir. Şekil 4'te replika edilmiş aynı dosya için farklı birer AVSG'ye sahip iki istemcinin durumu gösterilmektedir. Aşağıdaki Şekil 4'te, istemci B'nin bağlantısı ağın kırılmasından dolayı kopmuştur.



Şekil 4. Replika edilmiş aynı dosya için farklı birer AVSG'ye sahip iki istemci.

CODA'da replikasyon stratejisi olarak, **Read-One-Write-All (ROWA)** kullanılmaktadır. Bu yaklaşım, kopyalamanın yapıldığı tüm replikalarda şu anki verinin hepsinin bulunabilmesi olasılığını artırır [12].

3.2.3 Hataya Karşı Dayanıklılık

CODA'da eğer AVSG boş ise, istemcinin bağlantısını kopardığı söylenmektedir. Bu durumda hacmin bir kopyasında bulunan sunucuların herhangi birine istemci erişememektedir. Bu erişilebilirlik açısından oldukça büyük sorun teşkil etmektedir [15]. Sunucu replikasyonunun aksine bağlantı kopartma esnekliği replikasyon protokollerinin performans kaybı olmadan sağlanılmaktadır [12].

Bir istemci kasten ağdan bağlantısını kopartarak, bağlantı kopartma işlemine gönüllü olarak girebilir. Kullanıcı büyük bir disk önbelleğine sahipse, belirli bir süre boyunca CODA sunucusundan izole olarak çalışabilir. Zaman zaman, kullanıcı ağdan çıkıp tekrar girmek isteyebilir, bu anlarda yaptığı değişiklikleri CODA sunucularına yayımlaması gerekir [12]. Ana sorun, bir istemcinin dosyaların önbelleklenmiş kopyalarının üzerinde bağlantı koparıldıktan sonra da işlem yapabileceğinden emin olunmasıdır. CODA istemcileri stoklama (hoarding) denilen bir strateji kullanır, buna göre istemciler hala bağılyken önceden önbellek doldurulur. Stoklama fazı boyunca, istemci sürekli kendi önbelleğini güncellemek için çalışır. Bağlantı koparılması üzerine, istemci *öykünüm* (emulation) fazına girer. Bu durumda tüm işlemler yerel önbelleklenmiş kopyalar üzerinden yapılır. CODA, güvenli kanal üzerinden güvenli RPC ile iletişim ve sistem seviyesi yetkilendirme, erişim listelerini destekler [13].

3.3 Diğer Yaygın Dağıtık Dosya Sistemlerine Genel Bakış

Yukarıda bahsedilen sistemlerin haricinde bazı önemli dağıtık dosya sistemleri de bulunmaktadır. Bunlar genel yapısal özellikleriyle aşağıda incelenmiştir.

3.3.1 Plan 9, XFS Çalışma Mekanizması ğ İşletim Sistemleri'nin 1980'lerde Kazandığı

popülerliğe tepki olarak, **Plan 9**'un geliştirilmesine başlanmıştır. Plan 9, bir dosya-tabanlı dağıtık sistem değil bir dağıtık dosya sistemidir. Plan 9'da kaynaklar için dosya benzeri arayüzler önerilmiştir. İstemci-sunucu ayrımı Plan 9'da tam olarak açıklığa kavuşturulmamıştır. Plan 9'da CODA'nın aksine, **Write-Once – Read-Many (WORM)** stratejisini uygulayan düşük seviyeli katman bulunur. Ağ üzerinden iletişim 9P adı verilen standart bir protokolle sağlanılır [16]. 9P güvenli transport protokolü üzerinden çalışır. Plan 9 hakkında daha detaylı bilgiler Pike vd. 'nin yaptığı çalışmada [17] ve Bell laboratuvarları'nın dökümanında bulunabilir [18].

XFS genel yapı olarak, sunucusuz tasarıma sahiptir, tüm dosya sistemi birden çok makine arasında dağıtılmıştır [19]. Bu makineler istemcilerdir. Bu yaklaşım, merkezi bir organizasyona sahip olan dosya sistemlerinin aksine bir yaklaşımdır. AFS ve CODA bu tarz merkezi yaklaşıma girer. XFS ve xFS isimli iki farklı dağıtık dosya sistemi bulunmaktadır. Çalışmamızda sadece Anderson vd.'nin çalışmasında [19] belirtilen ve Berkeley NOW projesinde [20] geliştirilen dağıtık dosya sistemi XFS ile ilgilenilmiştir. XFS'in ana amacı, yüksek dereceden ölçeklenebilirlik ve hataya dayanıklılık sağlamaktır. XFS mimarisinde, üç farklı tipte süreç bulunur. Bir **depolama sunucusu**, bir dosyanın belirli kısımlarını depolamaktan sorumludur. Bir **üstveri yöneticisi**, bir dosya veri bloğunun gerçekte nerede depolandığının takip edilmesinden sorumludur. Bir **istemci**, XFS'de dosyalar üzerinde çalışmak için kullanıcı isteklerini kabul eden bir süreçtir. XFS'de önceleri tüm iletişim RPC mesajlaşmasıyla olurken daha sonra bu yaklaşım terk edilerek (performans kaybı nedeniyle) **aktif mesaj** yaklaşımına geçilmiştir [19,20].

4. Güncel Dağıtık Dosya Sistemleri Örnekleri Aşağıda bir kaç güncel dağıtık dosya sistemi tanıtılmaya çalışılmıştır.

4.1 OpenAFS, GoogleFS ve Microsoft DFS
OpenAFS, yukarıda anlatılan AFS sistemini temel alan açık-kaynak kodlu bir dağıtık dosya sistemi gerçekleştirmidir. AFS ilk öncele-ri CMU tarafından geliştirilirken sonrasında Transarc firması (şu anda IBM Pittsburgh Laboratuvarları) tarafından geliştirilmeye devam edilmektedir. OpenAFS geniş çapta UNIX, Linux, MacOS X ve Microsoft Windows sistemlerini destekler [21]. AFS, sunucular üzerindeki yükün azaltılmasını, büyük boyutlu chunk'lar veya bir bütün dosyanın önbelleklenmesi yoluyla sağlamaya çalışır. Bunun için ölçeklenebilir bir ortam yaratır. AFS ve OpenAFS dosya sistemi tüm istemciler için tek-düze bir isim uzayı yaratılır. Replikasyon'un sınırlı bir biçimi de desteklenmektedir.

Google Dosya Sistemi (GoogleFS), Google firması tarafından yüksek depolama istemlerini en verimli yoldan (büyük miktarda veri işlemek için) elde edebilmek adına tasarlanan ölçeklenebilir dağıtık dosya sistemidir. Ücretsiz olarak dağıtılmamaktadır. Bu sistem veri-yoğun uygulamaların büyük ölçekte kullanımını hedeflerken [21] hataya karşı dayanıklılığı da sağlamaya çalışır. Önceki dağıtık dosya sistemlerinin ana amaçları GoogleFS'de de mevcuttur. GoogleFS, snapshot ve kayıt ekleme işlemlerini içerir. *Snapshot*, bir dosya veya dizin ağacının en düşük maliyetle bir kopyasının yaratılmasıdır. *Kayıt ekleme* ise, her bir tekil istemcinin eklenmesinin garanti edildiği durumda, aynı dosyaya eşzamanlı olarak veri ekleyen birden çok istemciye izin vermektir [22].

Bir GoogleFS kümesi, tek bir master ve birden çok chunk sunucu bulundurur ve birden çok istemci tarafından erişilebilir durumdadır. Master sunucu tüm dosya sisteminin üstveri'sini tutar. Bu üstveri isim uzayları, erişim kontrol bilgileri, chunk'lara olan haritalama ve chunk'ların şu an ki konumunu içerir. Bu sayede, kullanılmayan chunk'ları çöp olarak toplama, chunk zaman sınırlı kiralama

yönetimi ve chunk sunucular arası chunk göçü mümkün olur. Master sunucu periyodik olarak her bir chunk sunucusuyla HeartBeat (Kalp atışı) ismi verilen mesajlarla iletişime geçerek, onların durumlarını kontrol eder. Bu yöntemler diğer kümeleme dosya sistemleri örnek alınarak oluşturulmuştur [23].

Microsoft firmasının geliştirdiği dağıtık dosya sistemi olan **Microsoft DFS (MS-DFS)** ise, örneğin istemci ve sunucu servislerinin bir kümesi olarak birçok dağıtık dosya paylaşımlarını organize ederek bunu bir dağıtık dosya sistemine dönüştüren bir yapıdır. Ücretsiz olarak dağıtılmamaktadır. MS-DFS yerleşim saydamlığını, hatanın varlığı veya birden çok paylaşımın olduğu durumlardaki yoğun yük altında veri elde edilebilirliğinin iyileştirilmesini amaçlar. MS-DFS, DCE/DFS ile karıştırılmamalıdır [24].

Windows 2000 ve Windows Server 2003'te DFS gerçekleştirmesi için iki yol bulunmaktadır. Birinci olarak; *tek başına* (standalone) bulunan DFS root'ların var olduğu durum ve ikinci olarak; *domain-tabanlı* DFS root'ları bir aktif dizin içerisinde bulunduğu durumdur. Dosya ve root bilgisi Microsoft Dosya Replikasyon Servisi (MS-FRS) ile replika edilir [25].

5. Dağıtık Sistemlerin Yapısal Karşılaştırılması

Çalışmamızda bahsedilen yedi adet dağıtık dosya sistemini belirli başlıklar altında incelemek mümkündür.

5.1 İsimlendirme ve Senkronizasyon

Dağıtık dosya sistemleri az veya çok bir biçimde aynı adres uzayını ve bağlanmayı desteklemektedirler. Daha önemli olan kısım ise isim organizasyonudur. Temelde iki yaklaşım bulunur. İlk yaklaşımda, (NFS ve Plan 9'da izlenen) her bir kullanıcının kendi özel isim uzayını almasıdır. Bunun tek eksiği, isim tabanlı

olarak dosya paylaşımının zor hale gelmesidir. İkinci yaklaşımda ise, bir global paylaşımli isim uzayı bulunur (CODA ve XFS'teki gibi). Tüm böyle sistemlerde, her bir kullanıcı özel bir yerel isim uzayı ile global isim uzayına eklenti yapabilme yeteneğine sahiptir.

GoogleFS de yerleşimden bağımsız bir isim uzayını ve yük dengesi-hataya dayanıklılık için verinin saydam olarak taşınabilmesine olanak sunar. NFS oturum semantik'i sağladığı için, sunucu tarafından sadece bir dosyayı kapatan en son sürecin yaptığı güncelleme hatırlanmaktadır. CODA'da, işlemsel semantik bulunur. Plan 9 dosyalar üzerinden işlem yaptığı ve UNIX semantiklerini sağladığı için bu işlemler dosya sunucusu üzerinden yapılır. XFS'de bu semantikler kendi

önbellekleme mekanizması yüzünden sağlanmakta ve bir dosya bloğunun şu anki sahibi yazma işlemini gerçekleştirmektedir.

5.2 Önbellekleme ve Hataya Dayanıklılık

Dağıtık dosya sistemlerinin çoğu istemci- taraflı önbelleklemeyi destekler. Sadece Plan 9, yazma işlemine dayalı önbellek tutarlılık protokolünü kullanır. Bu sayede her bir yazma işlemi hemen sunucuya iletilmiş olur. Diğer sistemler, önbelleği temizlemeden önce bir seri yazma işleminin (geriye-dönük yazım) önbellekler üzerinde gerçekleştirilmesine izin verirler. NFS v4'te bir dosya kapatıldığında değişiklikler sunuculara yayımlanacak şekilde düzenleme yapılmıştır. XFS dosya veri bloklarını önbellekler. CODA, tüm dosyanın önbelleklenmesini destekler.

Konu	NFS	CODA	Plan 9	XFS	OpenAFS	GoogleFS	MS-DFS
Tasarım amacı	Erişim saydamlığı	Yüksek elde edilebilirlik	Tekdüzelik	Sunucusuz sistem	Yüksek elde edilebilirlik	Veri depolama ve kümeleme	Yüksek elde edilebilirlik
Erişim modeli	Uzak	Upload/Download	Uzak	Log-tabanlı	Upload/Download	Seri numara ile kiralama	Uzak
İletişim	RPC	RPC	Özel	Aktif mesaj	RPC	Kalp atışı mesajı	RPC
Bağlanma büyüklüğü	Dizin	Dosya sistemi	Dosya sistemi	Dosya sistemi	Dosya sistemi	Chunk	Dizin
İsim uzayı	İstemci başına	Global	Süreç başına	Global	Global	Yerleşim bağımsız	Yerleşim bağımsız
Replikasyon	Minimal	ROWA	Yok	Striping	ROWA	Master veya chunk replikasyonu	FRS
Hataya dayanıklılık	Güvenli iletişim	Replikasyon ve önbellekleme	Güvenli iletişim	Striping	Replikasyon ve önbellekleme	Hızlı kurtarma ve replikasyon	Replikasyon
Paylaşım semantik	Oturum	İşlemsel	UNIX	UNIX	İşlemsel	UNIX-POSIX benzeri	UNIX benzeri
Önbellekleme birimi	Dosya (NFS v4'te)	Dosya	Dosya	Blok	Dosya	Chunk	Dosya
Erişim kontrolü	İşlemler ile	Dizin işlemleri	UNIX tabanlı	UNIX tabanlı	Dizin işlemleri	UNIX tabanlı	İşlemler ile

Tablo 1. Dağıtık dosya sistemlerinin karşılaştırılması.

AFS'nin aksine, GoogleFS bir dosya verisini depolama sunucularına XFS gibi dağıtır. GoogleFS, herhangi bir önbelleklemeyi dosya sistemi arayüzünün altında sağlamaz. MS-DFS, isim uzayında GoogleFS gibi yerleşim bağımsız bir yapı sergilerken bağlanma büyüklüğü dizin seviyesinde bulunur ve RPC iletişim yapar. Aksine GoogleFS, HeartBeat (Kalp atışı) denilen bir mesaj türü kullanılmaktadır. MS-DFS, replikasyon (sunucuları kopyalama) için MS-FRS denilen altyapıyı kullanılmaktadır.

5.3 Güvenlik

NFS v4'te güvenli kanalların gerçekleştirimi için standart bir arayüz var olan birçok güvenlik sistemi tarafından erişilebilir biçimde sağlanılır. NFS gerçekleştirimine bağlı olarak hangi mekanizmanın kullanılacağına karar verilir. CODA ve Plan 9 güvenli kanalları sağlarlar. Güvenilmeyen makinelerin XFS sistemine erişimine izin vermek için özel ölçütler belirlenmelidir. NFS v4'te kapsamlı bir işlemler listesi erişim kontrolü için kullanılır. Bu sayede esneklik sağlanılır. CODA, değişik bir yaklaşım sergiler, erişim kontrollerini sadece dizinler üzerinde yapılan işlemler için idare eder. Plan 9 ve XFS ise standart UNIX dosya izin ve çalıştırma işlemlerini takip etmektedir.

Tablo 1 'de yukarıda bahsedilen bazı dağıtık dosya sistemlerinin çeşitli kriterlere göre karşılaştırılması görülmektedir.

Tabloda kullanılan bazı kısaltmalar: **FRS** : *Dosya Replikasyon Servisi*, **ROWA** : *Read-one, Write-all yaklaşımı*, **RPC** : *Uzak Yordam Çağrısı*.

6. Sonuçlar

Günümüzde dağıtık dosya sistemleri, küçük ölçekli projelerden başlayarak, büyük çaplı ve görece büyük kapasite isterlerine sahip projelerde müşteriler veya istemcilerin güvenlik/performans ve yüksek elde edilebilirlik ve

hataya dayanıklılık isterlerini karşılamakta yoğun olarak kullanılmaktadır. Önbellekleme, sunucuya bağımlılığın azalmasıyla daha fazla hataya dayanıklı bir yapı oluşmasına yardımcı olur.

7. Kaynaklar

- [1]. Svobodova, L., (1984), "File servers for network-based distributed systems", ACM Computing Surveys, 16(4):353-398, Dec. 1984.
- [2]. Swinehart, D., McDaniel, G., Boggs, D., (1979), "WFS: A simple shared file system for a distributed environment", In Proc. of the 7th ACM Symp. on Oper. Syst. Princ., pages 9-17, Dec. 1979.
- [3]. CODA (CMU) websitesi, (2007), (Çevrimiçi: <http://www.coda.cs.cmu.edu/ljpaper/lj.html>).
- [4]. Wikipedia.org web sitesi-(2007), (Çevrimiçi: http://en.wikipedia.org/wiki/List_of_file_systems).
- [5]. Tannenbaum, S. A., Van Steen, M., (2002), "Distributed Systems Principles and Paradigms", Prentice Hall, ISBN-0-13-088893-1.
- [6]. Callaghan, B., (2000), "*NFS Illustrated*", Reading, MA: Addison Wesley 2000, Pages: 192,576,587.
- [7]. Kon, F., (1996), "Distributed File Systems Past, Present and Future A Distributed File System for 2006", Tech. Report, Uni. of Illinois at Urbana-Champaign, 1996.
- [8]. Srinivasan, R., (1995), "RPC: Remote Procedure Call Protocol Specification Version2", RFC1831, Aug.1995, p. 581.

- [9]. Bloomer, J., (1992), “Power Programming with RPC”, Sebastopol, CA, O’Reilly & Associates.
- [10]. Juszczak, C., (1990), “Improving the Performance and Correctness of an NFS Server”, Proc. Summer Techn. Conf., USENIX, pp53-63, [11]. Howard, J., vd., (1988), “Scale and Performance in a Distributed File System”, ACM Trans. Comp. Syst., vol. 6, no.2, pp 55-81, Feb. 1988.
- [12]. Satyanarayanan, M., Kistler, J., vd., (1990), “Coda : A Highly Available File System for a Distributed Workstation Environment”, IEEE Trans. Comp., vol. 39, no. 4, pp 447-459, Apr. 1990.
- [13]. Satyanarayanan, M., Siegel, E., (1990), “Parallel Communication in a Large Distributed System”, IEEE Trans. Comp., vol.39,no.3,pp328-348, Mar 1990.
- [14]. Satyanarayanan, M., (1990), “Scalable, Secure and Highly Available Distributed File Access”, IEEE Computer, vol.23, no.5, pp 9-21, May 1990.
- [15]. Satyanarayanan, M., (1992), “The Influence of Scale on Distributed File System Design”, IEEE Trans. Softw. Eng., vol. 18, no. 1, pp 1-8, Jan. 1992.
- [16]. Lampson, B., Abadi, M., Burrows, M., Wobber, E., (1991), “Authentication in Distributed Systems: Theory and Practice”, Proc. 13th ACM Symp. on Op. Sys. Princ., Asilomar, 1991, pp. 165-182.
- [17]. Pike, R., Presotto, D., Dorward, S., Flandrena, B., vd., (1995), “Plan 9 from Bell Labs”, Comp. Syst., vol. 8, no. 3, pp 221-254, Summer 1995.
- [18]. Bell Labs Comp. Sci. Research Center, (2000), “Plan 9 programmer’s Manual”, Vol. 1., Bell Labs, Lucent Technologies, Murray Hill, NJ, 3rd ed., 2000.
- [19]. Anderson, T., Dahlin, M., Neefe, J., Rosell, D., vd., (1996), “Serverless Network File Systems”, ACM Trans. Comp. Syst., vol. 14, no. 1, pp 41-79, Feb. 1996.
- [20]. Anderson, T., Culler, D., Patterson, D., The NOW Team, (1995), “A Case for NOW”, IEEE Micro, vol. 15, no. 2, pp 54-64, Feb. 1995.
- [21]. OpenAFS web sitesi. (Çevrimiçi : <http://www.openafs.org/>).
- [22]. Ghemawat, S., Gobioff, H., Leung, S., (2003), “The Google File System”, 19th ACM Symp. on Oper. Syst. Princ., Lake George, NY, Oct., 2003, (Çevrimiçi: <http://labs.google.com/papers/gfs.html>).
- [23]. Arpaci-Dusseau, R.H., Anderson, E., Treuhaft, N., vd., (1999), “Cluster I/O with River: Making the fast case common”, In Proc. of the Sixth Wrkshp on I/O in Paral.&Dist. Syst. (IOPADS ’99), pages 10–22, Atlanta.
- [24]. Microsoft DFS Resmi Web Sitesi, (2007), (Çevrimiçi: <http://www.microsoft.com/windowsserver2003/technologies/storage/dfs/default.mspx>).
- [25]. Microsoft DFS – Wikipedia.org Web Sitesi, (2007), (Çevrimiçi : http://en.wikipedia.org/wiki/Distributed_File_System_%28Microsoft%29).